In the Claims

1. (Currently amended) A method for outputting data from a legacy computer system, the data output in Extensible Markup Language format, the method comprising:

generating a model of the legacy computer system, the model comprising one or more incidents within one or more applications of the legacy computer system that output data;

mapping the model of the legacy computer system to an Extensible Markup Language schema; and

based at least on the mapping of the model of the legacy computer system to the Extensible Markup Language schema, automatically modifying the one or more applications of the legacy computer system that output data, the one or more modified applications operable to output data written using a Document Object Model from the legacy computer system in Extensible Markup Language.

2. (Previously presented) The method of Claim 1 wherein automatically modifying the one or more applications further comprises:

providing the legacy computer system with a writer engine, the writer engine having the Extensible Markup Language Schema loaded as a data file; and

calling the writer engine with the modified one or more applications when the one or more applications output data, the writer engine populating the Document Object Model according to the Extensible Markup Language schema by building a Document Object Model instance with one or more contexts.

3. (Previously presented) The method of Claim 2 further comprising:

applying one or more XSLT stylesheets to restructure the Document Object Model instance for outputting data in a predetermined format.

4. (Previously presented) A system for outputting data from a legacy computer system in an Extensible Markup Language format, the system comprising:

a modeling engine in communication with the legacy computer system, the modeling engine operable to generate a model of outputted data written by an application residing on the legacy computer system;

a mapping engine in communication with the modeling engine, the mapping engine operable to generate a modification specification by mapping the model of the outputted data written by the application residing on the legacy computer system to an Extensible Markup Language schema; and

a code generation engine in communication with the mapping engine and the legacy computer system, the code generation engine operable to modify code of the application residing on the legacy computer system, based at least on the generated modification specification, to directly output data from a Document Object Model as Extensible Markup Language.

5. (Previously presented) The system of Claim 4 further comprising:

a context table associated with the legacy computer system, the context table providing the Extensible Markup Language schema to the legacy computer system; and

a writer engine loaded on the legacy computer system and having the Extensible Markup Language schema stored as a data file, the writer engine communicating with the modified legacy computer system application to buffer data in plural contexts within the Document Object Model for output as Extensible Markup Language.

6. (Original) The system of Claim 5 wherein the writer engine is coded in the computer language of the legacy computer system.

7. (Previously presented) A method for outputting data as Extensible Markup Language from an application running on a computer system, the method comprising:

establishing a relationship of the output data and one or more Extensible Markup Language Document Object Model contexts, the output data corresponding to a write operation of the application for outputting the data, the one or more Extensible Markup Language Document Object Model contexts identifying a position in a target Extensible Markup Language schema of the output data corresponding to the write operation of the application;

building a Document Object Model instance with the one or more Extensible Markup Language Document Object Model contexts; and

outputting the data from the Document Object Model instance as Extensible Markup Language.

8. (Previously presented) The method of Claim 7 wherein establishing the relationship further comprises:

activating plural contexts simultaneously to buffer data for output as a complete Document Object Model instance.

9. (Previously presented) The method of Claim 8 wherein establishing the relationship further comprises:

creating a node for the output data; and

ensuring the correct cardinality of the created node.

10. (Previously presented) The method of Claim 7 further comprising:

generating the output data with the application;

calling a writer engine with the application when the write operation corresponding to the output data is executed;

providing the generated output data to the writer engine;

outputting, from the writer engine, data from a Document Object Model instance according to the target Extensible Markup Language schema.

11. (Original) The method of Claim 10 wherein the application comprises a legacy computer system application.

12. (Original) The method of Claim 11 wherein the writer engine comprises an application run in the computer language of the legacy computer system application.

13. (Previously presented) A system for outputting data from a Document Object Model as Extensible Markup Language, the system comprising:

a computer system having an application that outputs data, each data output instance corresponding to a write operation of the application; and

a writer engine loaded on the computer system and interfaced with the application, the writer engine having an Extensible Markup Language schema as a data file and the writer engine operable to write the data output by the application in plural active contexts;

wherein the application calls the writer engine when the application outputs data, the writer engine operable to build a Document Object Model instance for output of the data in accordance with the Extensible Markup Language schema.

14. (Previously presented) The system of Claim 13 wherein the writer engine populates a Document Object Model as a schema element aligned with a current one of the plural contexts by creating Extensible Markup Language tagged nodes down through the schema element of the output data if the schema element of the output data is a descendant of the current context.

15. (Previously presented) The system of Claim 14 wherein the writer engine is further operable to determine a minimal mutual ancestor of the schema element of the output data and the current context and to traverse the Extensible Markup Language tagged nodes for the current context up to the minimal mutual ancestor and to create Extensible Markup Language tags for the schema element of the output data down from the mutual ancestor.

16. (Previously presented) The system of Claim 13 wherein the computer system comprises a legacy computer system.

17. (Previously presented) The system of Claim 16 wherein the application comprises a legacy computer system application modified to output an Extensible Markup Language schema element with the output data.

18. (Original) The system of Claim 17 wherein the writer engine is written in the code of the legacy computer system.

19. (Original) The system of Claim 18 wherein the code comprises COBOL.

20. (Previously presented) A method for outputting data from a legacy computer system from a Document Object Model instance as Extensible Markup Language, the method comprising:

modifying an application of the legacy computer system such that the modified application is operable to output data having a schema element of a target Extensible Markup Language schema, the output data corresponding to a write operation of the application;

outputting data from the modified application, the output data having the schema element of the target Extensible Markup Language schema;

aligning the schema element of the output data and a current context;

writing the schema element of the output data to a current one of plural contexts of the target Extensible Markup Language schema; and

populating a Document Object Model with the output data to output an Extensible Markup Language instance.

21. (Previously presented) The method of Claim 20 wherein aligning the schema element of the output data further comprises:

determining that the schema element of the output data is a descendant of the current context; and

creating appropriate Extensible Markup Language tags down through the schema element of the output data, each Extensible Markup Language tag down through the schema element of the output data being associated with an ancestor of the schema element of the output data.

22. (Previously presented) The method of Claim 21 wherein aligning the schema element further comprises:

determining a minimal mutual ancestor of the schema element of the output data and the current context;

traversing the Extensible Markup Language tags for the current context up to the mutual ancestor of the schema element of the output data and of the current context; and

creating the Extensible Markup Language tags for the schema element of the output data down from the mutual ancestor to the schema element of the output data.

23. (Previously presented) A method for modeling a legacy computer system, comprising:

identifying incidents ~~of~~ within one or more applications of the legacy computer system that output data;

associating the identified output incidents within the one or more applications with an Extensible Markup Language schema;

defining a control flow graph of the identified output incidents;

based at least on the association of the identified output incidents within the one or more applications with the Extensible Markup Language schema, creating a modification specification for modifying the one or more legacy computer system applications to provide output from a Document Object Model instance as Extensible Markup Language; and

automatically modifying, based at least on the modification specification, the one or more legacy computer system applications such that the one or more modified legacy computer system applications provide output from the Document Object Model instance as Extensible Markup Language.

24. (Canceled).

25. (Previously presented) A system for modeling an application of a legacy computer system, the application operable to output data, comprising:

a modeling engine interfaced with the legacy computer system, the modeling engine operable to:

analyze the application loaded on the legacy computer system to identify incidents within the application that output data from the legacy computer system; and

generate a control flow graph of the output operations within the applications, the control flow graph having plural nodes, each node associated with an output incident;

a graphical user interface in communication with the modeling engine, the graphical user interface operable to display the control flow graph and the incidents within the application that output data from the legacy computer system, wherein the graphical user interface maps the incidents of the application with the control flow graph and an Extensible Markup Language schema; and

a code generation engine in communication with the graphical user interface and the legacy computer system, the code generation engine operable to modify, based at least on the mapping of the incidents of the application with the Extensible Markup Language schema, legacy computer system application code to directly output data from a Document Object Model as Extensible Markup Language.